Open-Source Implementation of Medical Databases

Mohamed I. Owis
Department of Systems and Biomedical Engineering, Faculty of Engineering, Cairo University, Giza, Egypt.
E-mail: miowis@k-space.org

Abstract- **Clinicians use of information systems and the ability of these systems to share patient data are two major steps towards the modernization of the health care systems. However, conventional software presents many difficulties to achieve this goal; among them how to share patient information and the proprietary nature of the IT systems. This made many health care facilities consider open source software as a substitute to proprietary systems because, among other reasons, it allows different IT systems to run compatibly, it is available freely or at minimal cost and it gives more flexibility. I hereby present my experience in reimplementing an image database using open-source components.**

Keywords- **Open-source, ORDBMS, biomedical database**

I. Introduction

During the last two decades, the role of information technology (IT) in health care has developed from being an administrative tool for billing and accounting to becoming an important clinical tool for improving the quality and efficiency of health care delivery. In the 1970s and 1980s, IT helped large health care providers handle their business by automating financial records, payroll, and other back-office tasks. In the early 1990s, information systems began playing a role in departments, such as laboratory, radiology, and pharmacy, and provided tools for nursing tasks, such as bed management. Starting in the late 1990s, health care providers began to shift focus from automating individual departments to building integrated-care delivery systems, which required that health care facilities also integrate and share the different IT systems that they use. However, because conventional software presents many obstacles to achieving this goal, a number of health care IT leaders now are considering open source software as an alternative because, among other reasons, it allows different IT systems to operate compatibly and widely available at minimal cost, and gives health care providers more options and flexibility.

As an example of migrating clinical IT systems from proprietary to open source, in this paper I present my experience reimplementing a brain-image database (BRAID) using open source components. BRAID has been first implemented based on the proprietary object-relational database-management system (ORDBMS) Illustra [1]. Despite many advantages of implementing BRAID using an ORDBMS, including the definition of new data types and operators, the proprietary nature of this ORDBMS diminished the ability to maintain the database and to share software. To overcome these disadvantages and to facilitate sharing of BRAID's source code, a complete open-source software environment was needed. In addition to the DBMS, this environment includes the operating system, and the web-based interface to the data. We chose the freely available hypertext preprocessor PHP (php: hypertext preprocessor, http://www.php.net) to generate BRAID's web pages and thus manage the user-DBMS interface. Similarly, we chose the open-source web server, Apache (http://httpd.apache.org/), to manage BRAID's web site. Finally, we chose to integrate these components under the Linux operating system (http://www.linux.org). An important benefit of our choice of open-source components is the provision for a database administrator, through access to all source code, to construct and to maintain a brain-image database similar to ours.

II. Open-Source Object Relational Database Management System

One of the principal goals of the BRAID project is the development of the computational infrastructure necessary to support the management and analysis of image-based clinical trials (IBCTs), in which medical imagery constitutes a critical component of the data collected during a clinical trial. Most IBCTs supported by BRAID involve lesion-deficit analysis; one goal of such analysis is to determine associations among locations of abnormal voxels and outcome measures or predisposing factors. For a particular study, the data may consist of 3-dimensional maps of subjects' lesions (delineated and registered to a common spatial standard), demographics, predisposing factors, and outcomes.

Based on the nature of the trials in BRAID, the DBMS needed for BRAID must allow to implement and enforce data formats for images and clinical information. Furthermore, a standard query language should allow performing complex statistical and visualization operations on these data, using a combination of clinical and image-based selection criteria.

There are three general DBMS architectures that we could use to implement BRAID: relational (RDBMS), object-relational (ORDBMS), and object-oriented (OODBMS) [2]; all three of these approaches have been used to manage image data. In the following paragraphs we describe relative advantages of RDBMSs, ORDBMSs, and OODBMSs for managing image data. Of the three approaches, RDBMS technology is the most mature, and is the most widely used. Examples of this approach include the Extensible Neuroimaging Archive Toolkit project [3], which was built using MySQL [4, 5]; and a molecular-biology information system [6], built using McKoi [7], a Java-based RDBMS. The principal disadvantage of this approach lies in the lack of native support for image data and statistical tools; for example, images are represented only as links to binary files.

Due to this deficiency, there have been efforts to add an object-oriented software layer (e.g., Perl) over a RDBMS to support data type extensions [8]. Although this approach supports extensibility, it has two principal problems. First, designing, implementing and maintaining an object layer would complicate code sharing and maintenance. Second, introducing an additional software layer would impair integration at the DBMS level.

In contrast to a RDBMS, both ORDBMS and OODBMS approaches allow the integration of new data types, and

`operators on these data types, into the DBMS. Several ORDBMSs are available; the University of California, Berkeley pioneered this approach with the development of postgres [9]. Postgres was the predecessor of Illustra, a commercial ORDBMS, and of PostgreSQL [10], an open-source ORDBMS. Independently, IBM developed Starburst [11], Informix developed Universal Server, and Oracle [12] included object-relational features into its DBMS products starting with Oracle 8.0. ORDBMSs have been employed extensively to manage image data; [13,14,15,16,17,18,19; http://www.bioimage.org]. These databases have been built using Postgres, Starburst, Oracle 9i, Illustra, Universal Server, and PostgreSQL. Similarly, the ability of OODBMSs to handle complex data has motivated the development of several additional systems [20], such as O2 [21].

In their review of suitable DBMSs for neuroimaging, Diallo et al. reported that OODBMSs have some disadvantages relative to ORDBMSs, including incompatibility with legacy DBMSs, and absence of a well-defined standard [22]. Stonebraker's four-quadrant classification of DBMSs [23] supported this conclusion; Stonebraker's scheme indicates that an ORDBMS may be the best choice when managing complex data and processing complex queries. Therefore ORDBMS was the chosen technology to implement BRAID rather than an OODBMS or a RDBMS, because an ORDBMS would afford the extensibility promised by an OODBMS, while maintaining the efficiencies and standards of a RDBMS.

Of the systems available at the time of the initial implementation of BRAID, Illustra seemed to be the most mature, and to have the most support. This initial implementation allowed defining several image data types (e.g., binary, integer, RGB), to support these data types with query-accessible image-processing and statistics operators. BRAID database has been subsequently used to manage and analyze data from several IBCTs [1]. However, the BRAID's reliance on proprietary software impeded sharing and maintenance of BRAID. We therefore opted to reimplement BRAID using PostgreSQL, a robust (and the only widely used) open-source ORDBMS (http://www.postgresql.org).

Almost all functions of the structured query language (SQL) standard are implemented in the query language of PostgreSQL as a full-function relational database. Moreover, all object-oriented features are included as new specific instructions or reserved keywords. The server can handle simultaneous queries, users and databases. Several clients and interfaces for PostgreSQL are available; we used two interfaces, psql and pgaccess (http://www.pgaccess.org), which are particularly important with respect to the design and maintenance of BRAID.

PostgreSQL is extended by adding operators written in C or Java to the system; software developers can thus query a database from within a program they have written in any of these languages. The PostgreSQL engine allows the user to develop new components, and add them to the kernel as dynamically linked libraries. These components, represented as simple functions or procedures, may consist of SQL statements or C functions. In our project, we defined an image data type, and image and statistical operators. Since we extended the kernel using dynamically linked libraries, this process did not require recompilation of the database kernel.

III. DATABASE DESIGN

A. *Data Model*

BRAID's database consists of data specific to each IBCT, in addition to a collection of anatomic atlases of reference brain-image volumes that describe the spatial extents of brain structures. We model an atlas with two entities: ATLAS and STRUCTURE. The ATLAS entity encodes spatial information for each atlas included in BRAID. The image of each atlas is color-coded to label different structures. The STRUCTURE entity represents information related to the atlas-defined brain structures. The main attribute of the STRUCTURE entity is *image*, which is defined as a binary mask in a particular coordinate space. The data for the STRUCTURE entity are extracted from the color-coded atlas images.

For a particular study, the data may consist of 3-dimensional maps of subjects' lesions (delineated and registered to a common spatial standard), demographics, predisposing factors, and outcomes. We therefore chose to model IBCT data with three entities: STUDY, SUBJECT, and SUBJECT_IMAGE. The STUDY entity captures information for a particular study (e.g., study name). The SUBJECT entity records information relevant to each individual (e.g., age). The SUBJECT_IMAGE entity captures information related to each subject's image data. This entity includes an attribute of the data type *image* to hold an image of a subject's lesions. In addition, we include an additional subject entity for each IBCT, such as STUDY1_SUBJECT; this entity models clinical data specific to the corresponding IBCT, whereas the generic subject entity models subject data, such as date of entry, that we expect to have for all IBCTs. The most suitable way to model these distinct clinical variables is by using the table-inheritance feature in PostgreSQL. Unfortunately, this feature is still incomplete, since PostgreSQL does not support inheritance of foreign keys [http://www.postgresql.org/docs/8.1/interactive/ddl-inherit.html]. For example, we will not be able to relate the study-specific table to the SUBJECT_IMAGE table, or otherwise to create an entity and table for images of each study separately, even though images have the same specifications across studies. Therefore, we utilized the concept of enhanced-ER (EER) modeling [2]. We don't need this for the SUBJECT_IMAGE entity since images of different IBCTs have common features that are captured in the *image* data type (explained below).

The EER model includes all of the modeling concepts of the ER approach; in addition, it includes the *subclass* and *superclass* concepts, which support inheritance of the general specification of the superclass to all subclasses. In BRAID, this condition obtains for the SUBJECT entity (a superclass) and the STUDY1_SUBJECT and STUDY2_SUBJECT entities (subclasses). Each subclass entity represents a specific clinical trial. The EER approach (or eventually inheritance) permits us to to add new IBCTs at any time, by generating an additional subclass entity (e.g. STUDY3_SUBJECT).

Finally, the PIV (precomputed intersection volume) entity serves as a connection among the principal components. It contains the volume of intersection of an atlas structure with a subject's lesions; each intersection volume is computed off-line to improve the efficiency of queries. In Fig. 1, we present

BRAID's entity-relationship (E-R) diagram. The relationships between entities are either one-to-one or one-to-many. Each of BRAID's seven entities is mapped to a table in our extended relational-database implementation. As expected for a relational database, every row within a table is uniquely identifiable through the primary key for that table. Relationships between entities are established through primary and foreign keys that guarantee data integrity and coherence. In addition, we created derived fields to improve query performance; for example, we precompute the volumes of intersection in the precomputed-intersection-volume (PIV) table, because computing volumes is relatively time consuming.

To map the subclass–superclass relationship onto tables, each study is assigned its own table. The data for a given study are stored as a single table in which columns represent clinical information and each row represents a subject of this IBCT. Each subject has a unique identifier that acts as a key. To incorporate data from a new IBCT, a database administrator (DBA) need complete only four steps: first, create a new tuple for this study in the STUDY table. Secondly, incorporate the basic information for the subjects of this new IBCT into the SUBJECT table, so that each subject has a unique identification number that is used to insert that subject's clinical and image data. Then, create a new table for this study's clinical data, whose fields are the clinical variables of this particular IBCT. Finally, insert subjects' image-data into the SUBJECT_IMAGE table.

B. *Data types*

Unlike image databases that store only file names and locations of images, BRAID stores image data using our defined data type *image*. PostgreSQL maintains the *image* data type in binary form when queried, and thus can pass these data efficiently to data-processing programs, such as image-processing and statistical operators.

Using the extensibility of PostgreSQL, we defined the *image* data type, which corresponds to a C-language structure. The key elements of the *image* data type are: modality (e.g., magnetic-resonance (MR), positron-emission tomography (PET)), value type (e.g., Boolean, integer, RGB). We use the Boolean type for a binary image mask that represents, for example, a single atlas structure. We use the integer type to represent a subject's lesion image, in which different integer

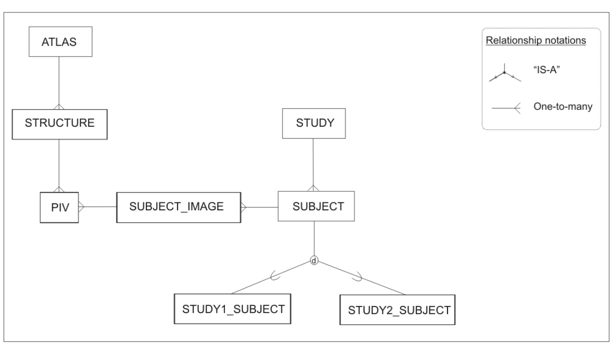

Figure 1. The entity-relationship diagram for the BRAID schema. A rectangle represents an entity, and a line represents a relationship between entities. For the subclass/superclass (IS-A) relationship, the "d" notation specifies the constraint that subclasses must be disjoint; i.e., an entity can be a member of at most one of the subclasses.

values may be assigned to different lesions to distinguish them. We use the RGB type to represent an atlas image, in which each atlas structure is color-coded. The value type is implemented as a C union, which facilitates extensibility. Another element of the image data type is format (e.g. zyxx, raw); this value indicates the format for the image data (see next element). For example, the zyxx format represents a line segment (z, y, x_1, x_2) that has a certain value (e.g., color, signal intensity, 0/1 binary value). Then the image data itself that encoded in to the format specified in the previous element. Finally some parameters such as acquisition date and voxel size in mm.

C. *Spatial operations*

To support the execution of image-based queries, we defined and implemented a set of operators, including intersection, sum, and threshold, for the *image* data type. We implemented the intersection procedure and mapped it to the "*" operator in PostgreSQL, which can be incorporated into a SQL statement. Similarly, we implemented the summation procedure and mapped it to the "+" operator.

We have also implemented other operations, such as *threshold,* which allows the user to apply different types of thresholds (integer or RGB values) to an image. The *threshold* function returns an *image* with only those voxels satisfying the threshold criterion. Another example is the *volume* function, which computes the volume of an *image*. We have also implemented functions for producing output: *save_image* and *generate_png*. For example, the function *save_image* saves an *image* to a file, and the function *generate_png* generates a portable network graphics (PNG) image (http://www.libpng.org/) from BRAID's internal image format.

D. *Statistical operations*

As described above, after image preprocessing, segmentation of lesions, and registration to a common standard, the structural image data, consisting of registered lesions, are inserted into the database. For each subject, these image data, combined with clinical variables, form the data to be analyzed or visualized. Note that, although BRAID is designed to work with registered images, the database is itself independent of which particular registration method was used.

Data-mining methods for these data operate on a resolution range, from spatially distinct atlas structures, to the voxel level [1]. BRAID supports simple univariate atlas-based analysis through an on-line query system, based on two statistical methods: chi-square and Fisher exact statistics. Voxel-based analysis is performed off-line, due to the high computational burden of mining thousands of voxel and clinical variables.

We implemented these two statistical tests as aggregate functions in PostgreSQL; an aggregate function computes a single result value from a set of input values (rows). For example, there are aggregates to compute the *count, sum, avg* (average), *max* (maximum) and *min* (minimum) over a set of rows. A statistical aggregate function in BRAID builds the contingency table by iteratively examining all subjects and structures to count binary values of (a) whether the state of a clinical variable for a subject meets a specific criterion; and (b) whether a structure (or voxel) is labeled abnormal. A structure is labeled abnormal if its volume of intersection with a subject's lesions exceeds a user-supplied threshold.

E. Data Security

We have implemented security primarily through de-identification of subjects' data and through the assignment of password-protected accounts for our collaborators. To protect subjects' privacy, our collaborating researchers strip all clinical and image data of potential identifiers, and give each subject a unique numerical ID, before sending the data to BRAID; this ID number, in conjunction with the study name, represents a key for that subject in the database. As an additional security measure we incorporate into BRAID only the subjects' registered lesions (as a binary-mask representing the spatial location of the lesions), rather than the original MR images, thus preventing intruders from reconstructing facial surfaces.

In addition, BRAID provides a user-level security model by assigning different levels of user access managed by the PostgreSQL server. BRAID's database administrator (DBA) has full permission on all database objects and is responsible for all-data manipulation tasks: populating new clinical or image data, and deleting or updating any datum. The DBA assigns for each IBCT a user or a group of users who will have read-only access (primarily through BRAID's web site) to that study's clinical and image data.

F. Data Entry

BRAID provides a set of off-line C programs (which connect to the PostgreSQL server) to populate the database with spatial data and clinical data from an IBCT. These procedures are run by BRAID's DBA, and are facilitated by psql and other database-management tools available in PostgreSQL.

IV. Conclusion and Future Work

We have described the design and open-source implementation of BRAID, a system for managing, querying, analyzing and visualizing brain MR images. We implemented BRAID using PostgreSQL, an open-source ORDBMS, extended to handle spatial data types and user-defined functions; this open-source implementation allows maintaining and to share our source code much more easily than working with a proprietary DBMS.

We identified and analyzed requirements for supporting image-based clinical trials. We introduced and implemented the *image* data type, along with image-processing and statistical operations within PostgreSQL. We will extend the *image* data type to include lesion type, to facilitate lesion-deficit analysis for different types of lesions.

We implemented and integrated simple statistical tests to support exploratory data-mining queries such as "list Fisher-exact p-values for the pair-wise combination of brain structures and clinical conditions" for a particular study. A further improvement towards a flexible exploratory environment would be to integrate statistical-analysis tools within BRAID. Consistent with our open-source approach, we favor the statistics package R (http://www.r-project.org/). There have been some efforts to integrate R into PostgreSQL, such as PL/R (http://www.joeconway.com/plr/).

In summary, we have found that open-source tools can provide a sound foundation for the construction of a stable, high-performance, extensible, scalable image database. We have also found that reliance on open-source components greatly facilitates sharing of our source code and techniques. We anticipate that an open-source statistics datablade, and support for inheritance, will further improve performance and the utility of this database for our colleagues performing image-based clinical research.

Acknowledgment

This work was supported by The Human Brain Project, National Institutes of Health grant R01 AG13743, which is funded by the National Institute of Aging, and the National Institute of Mental Health.

References

[1] Herskovits E.H., "An architecture for a brain-image database." *Methods of Information in Medicine*. 39 (4/5), pp. 291-297, 2000

[2] Elmasri R. and Navathe S., *Fundamentals of Database Systems*, 3rd ed., Addison-Wesley, 1999.

[3] Extensible Neuroimaging Archive Toolkit; http://arc.wustl.edu/xnat/

[4] MySQL database management system http://www.mysql.org

[5] Marcus D.S., Snyder A.Z., Williams L.E., O'Brien K.O., Morris J.C., Buckner R.L., "The Extensible Neuroimaging Archive Toolkit: Informatics Tools for Managing and Exploring Neuroimaging Data." *Program No. 428.5.2003. Washington DC: Society for Neuroscience*, 2003. Online.

[6] Miranker D, Xu W, Mao R., "MoBIoS: a Metric-Space DBMS to Support Biological Discovery." *15th International Conference on Scientific and Statistical Database Management (SSDBM 2003)*. Cambridge, Massachusetts: 241-244

[7] McKoi; http://mckoi.com/database/.

[8] Jakobovits R., "WIRM A Perl-based application server." *Web Techniques*. September, 97–100.

[9] postgres; http://www.postgresql.org/files/documentation/books/awpgsql/node9.html.

[10] PostgreSQL; http://www.postgresql.org/.

[11] Starburst; http://www.almaden.ibm.com/cs/ starwinds/starburst.html.

[12] Oracle; http://www.oracle.com

[13] Ogle V. E. and Stonebraker M. "Chabot: retrieval from a relational database of images." *Computer.* 28(9), 40-48, 1995.

[14] Arya M., Cody W., Faloutsos C., Richardson J., and Toga A. "A 3D medical image database management system." *Computerized Medical Imaging & Graphics.* 20(4), 269-284, 1996.

[15] Martone M. E., Zhang S., Gupta A., Qian X.; He H.; Price D. L., Wong M., Santini S., Ellisman M. H. "The Cell-Centered Database: A Database for Multiscale Structural and Protein Localization Data from Light and Electron Microscopy." *Neuroinformatics.* 1(4), 379-396,2003.

[16] Diallo B., Dolidon F., Travere J.-M., and Mazoyer B. "B-SPID: An Object-Relational Database Architecture to Store, Retrieve, and Manipulate Neuroimaging Data." *Human Brain Mapping.* 7,136–150,1999.

[17] Shahabi C. , Dashti A.E., Burns G., Ghandeharizadeh S., Ning Jiang, Swanson, L.W. "Visualization of spatial neuroanatomical data. Visual Information and Information Systems;" *Third International Conference, VISUAL'99.* p 801-8, 1999.

[18] Carazo JM. Stelzer EH. "The BioImage Database Project: organizing multidimensional biological images in an object-relational database." *Journal of Structural Biology.* 125(2-3), 97-102,1999.

[19] Shotton D. M. "The evolution of the BioImage Database: where we are going, why, and with whom?" *E-BioSci/ORIEL Joint Summer Meeting* 21-24 June 2002, Wiesloch, Germany.

[20] Fredriksson J. "Design of an Internet accessible visual human brain database system," *IEEE International Conference on Multimedia Computing and Systems*, Florence, Italy. 1. 469-474,1999.

[21] Deux O. "Story of O2." *IEEE Transactions on Knowledge and Data Engineering.* 2(1), 91-108, 1990.

[22] Diallo B., Travere J.-M., Mazoyer B. "A Review of Database Management Systems Suitable for Neuroimaging." *Methods of Information in Medicine*, 38(2), 132-139,1999.

[23] Stonebraker M. *Object-relational DBMSs: the Next Great Wave.* Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1996.